(12) UK Patent Application (19) GB (11) 2 304 209 (13) A

(43) Date of A Publication 12.03.1997

(21) Application No 9516082.6

(22) Date of Filing 04.08.1995

(71) Applicant(s)
Motorola Limited

(Incorporated in the United Kingdom)

European Intellectual Property Operation, Jays Close, Viables Industrial Estate, BASINGSTOKE, Hampshire, RG22 4PD, United Kingdom

(72) Inventor(s)
Colin MacDonald
John Wilson Ralston

(74) Agent and/or Address for Service
Christopher Stanislaw Hirsz
Motorola Limited, European Intellectual Property Operation, Midpoint, Alencon Link, BASINGSTOKE, Hampshire, RG21 7PL, United Kingdom

(51) INT CL^6
G06F 9/445

(52) UK CL (Edition O)
G4A AFL

(56) Documents Cited
GB 2184577 A **EP 0100140 A2**

(58) Field of Search
UK CL (Edition N) **G4A AFL**
INT CL^6 **G06F 9/24 9/445**
On-line : WPI, INSPEC, COMPUTER

(54) Starting up a processor system

(57) A processor system has a Central Processing Unit (CPU) (3), a Direct Memory Access Controller (DMAC) (25) coupled to the CPU, a non-volatile storage device (NVSD) (7) for storing at least initial operating code, Random Access Memory (RAM) (8) coupled to the CPU, and a start-up logic circuit (1) coupled to the DMAC. The start-up logic circuit enables the DMAC, when a start-up signal is received, to load Power-On Values from memories (28, 31 and 34) into operating registers (26, 29 and 32) to control the NVSD to load at least the initial operating code for the CPU into the RAM and, when at least part of the initial operating code is loaded in the RAM, to enable the CPU to start to access it from the RAM, thereby initiating start-up of the CPU.

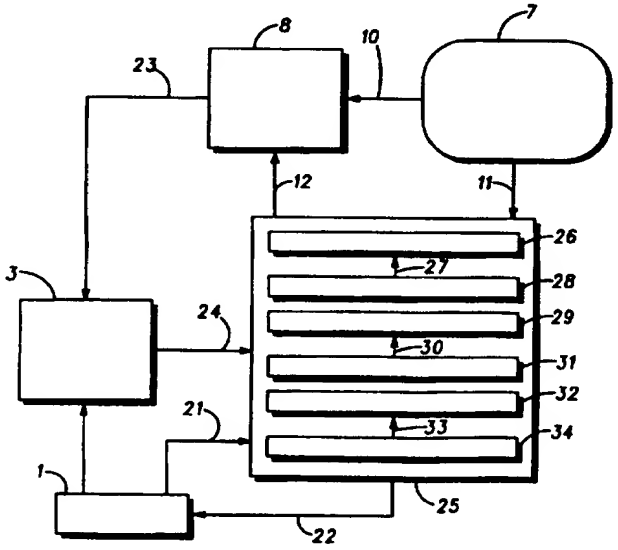

FIG.3

At least one drawing originally filed was informal and the print reproduced here is taken from a later filed formal copy.

GB 2 304 209 A

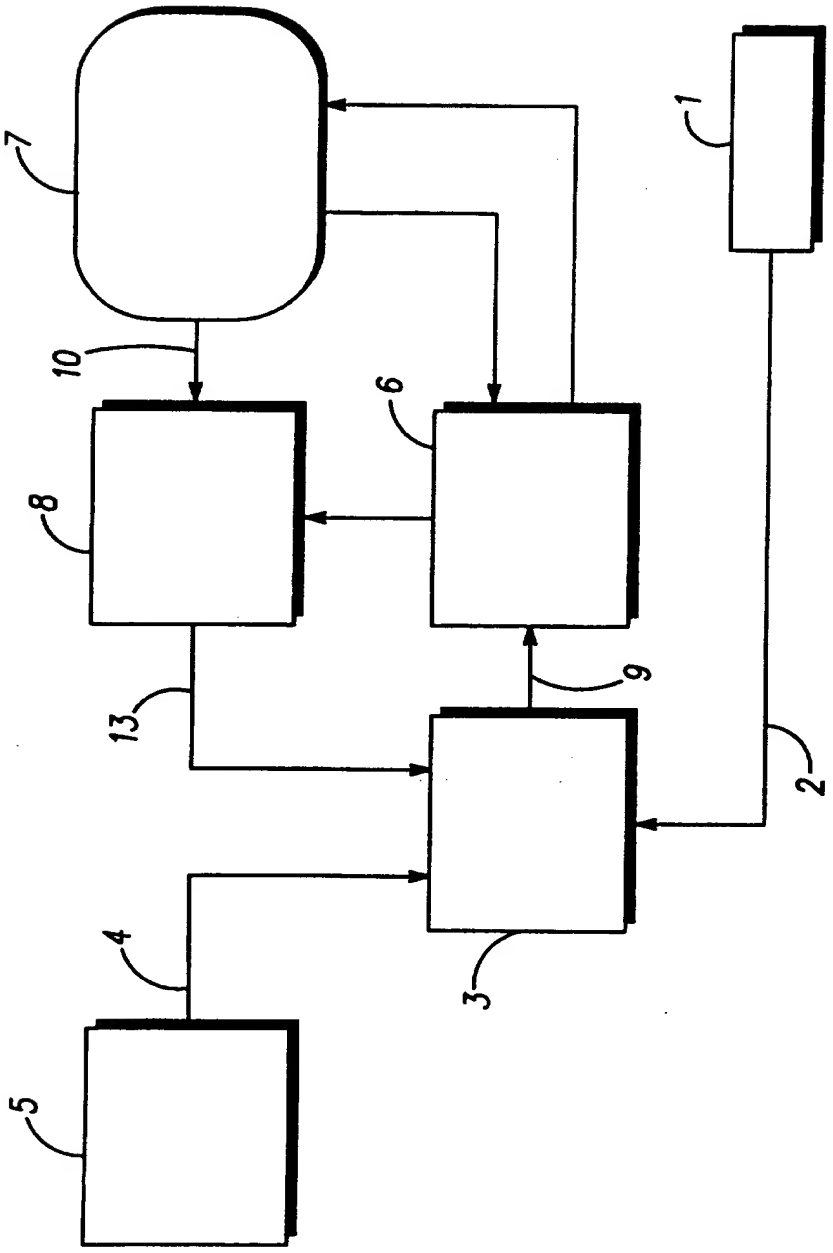

FIG.1
—PRIOR ART—

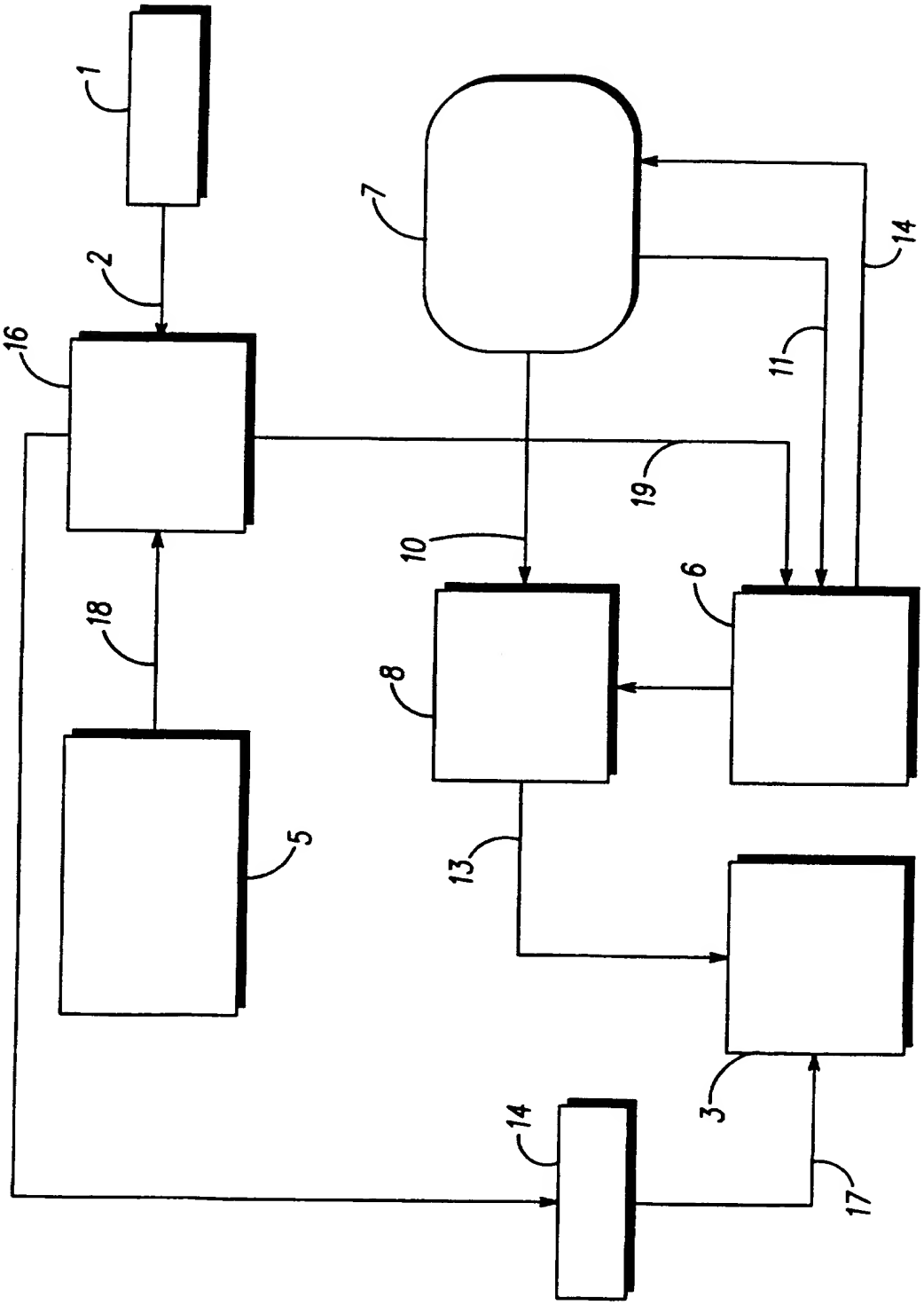

FIG.2
—PRIOR ART—

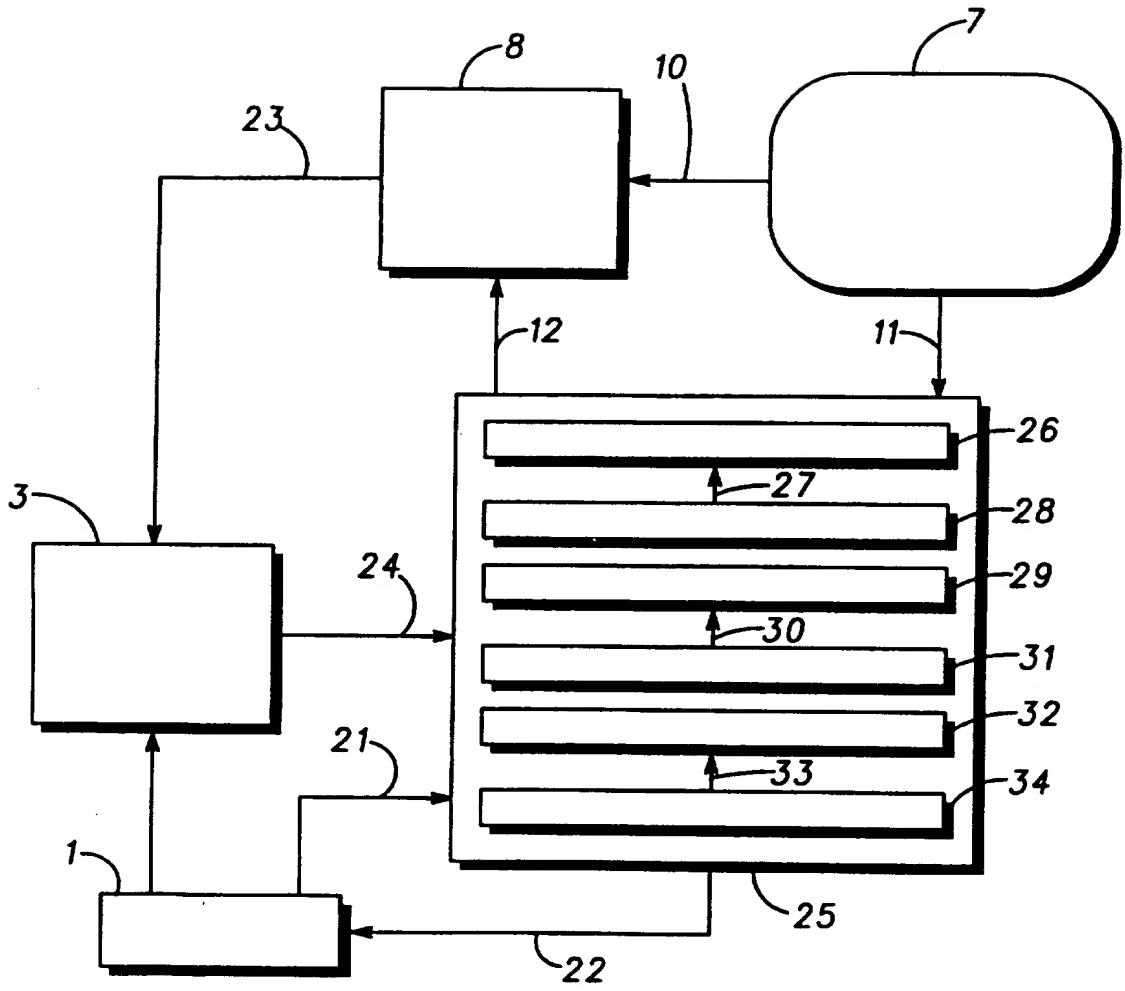

FIG.3

2304209

Processor System and Method of Starting-Up a Processor System

Field Of The Invention

This invention relates to a processor system, and particularly to a processor system having a Central Processing Unit (CPU) controlling a Direct Memory Access Controller (DMAC), which accesses a data storage device and writes the data into a volatile memory for access by the CPU, and to a method of starting-up such a system.

Background Of The Invention

In order for a processing system to function, the CPU must have access to operating software (OS). Since Non-Volatile Memory (NVM), such as Electically Programmable Read Only Memory (EPROM), is relatively expensive, the OS is generally stored on a peripheral data storage device and is accessed by a DMAC and loaded into a volatile Random Access Memory (RAM), from where it is read by the CPU. In high power processing systems, where the CPU may have a relatively large data width, for example 32 or 64 bits, the DMAC also performs the function of reformatting the data from the peripheral data storage device, into the correct data width for access by the CPU. It will be appreciated that the DMAC referred to herein is a basic hard-wired logic device with no processing power of its own.

However, in order to control the DMAC to perform the access, reformat and load operations described above, the CPU must first be made operational, or booted up, at least to a limited extent. This requires that, at start-up the CPU first accesses at least a small amount of operating code sufficient to enable to CPU to control the DMAC to allow more of the operating code to be accessed. Since, at start-up, the RAM has not yet been loaded, this initial operating code must be stored in NVM, which is usually EPROM provided on-chip with the CPU. Such EPROM is, however, expensive and is therefore usually of smaller data width than the CPU, which consequently requires longer time to access all the operating code than if the EPROM had the same data width as the CPU.

In order to remove the necessity for such EPROM, it is known for a second CPU to be provided to access the peripheral data storage device immediately when a restart signal is applied, reformat the operating code and store it in the RAM, from where the first CPU can access it efficiently. However, this requires a second CPU, which is clearly expensive, and does not solve the problem of how that second CPU is first started up. In this

case, although the second CPU is performing the function of a DMAC, it is not a simple DMAC as defined herein since it has processing capabilities itself.

It is therefore an object of the present invention to provide a processor system in which the CPU is provided with access to operating software at start-up, while mitigating the disadvantages of the known systems.

Brief Summary Of The Invention

Accordingly, the invention provides a processor system comprising a Central Processing Unit (CPU), a Direct Memory Access Controller (DMAC) coupled to the CPU, a non-volatile storage device for storing at least initial operating code, volatile Random Access Memory (RAM) coupled to the CPU, and a start-up logic circuit coupled to the DMAC , the start-up logic circuit enabling the DMAC when a start-up signal is received, the DMAC being configured to access the non-volatile data storage device to load at least the initial operating code for the CPU into the volatile RAM and, when at least part of the initial operating code is loaded in the volatile RAM, to enable the CPU to start to access at least part of the initial operating code from the volatile RAM, thereby initiating start-up of the CPU.

In a preferred embodiment, the start-up logic circuit is coupled to the CPU to restrain the CPU from accessing the volatile RAM when a start-up signal is received until the DMAC enables the CPU. The DMAC can enable to CPU directly, or the start-up logic circuit can enable the CPU when it receives a signal from the DMAC.

Although the non-volatile storage device is preferably a Non-Volatile Memory (NVM), which is preferably coupled to the DMAC, it could alternatively be a hard file, such as a disc, CD,or optical CD, or other storage medium, such as magnetic tape. The DMAC can access the non-volatile storage device either by a direct link or by indirect data transfer, such as wireless transfer, e.g. by radio, infra-red, or other means. The non-volatile storage device need not be of the same data width as the CPU. If it has a different data width, then the DMAC preferably reformats the initial operating code into the same data width as the CPU before it is loaded into the volatile RAM.

The non-volatile storage device can store complete operating software for the CPU, as well as the initial operating code, or the operating

software can be stored on other non-volatile storage devices, which can be accessed by the DMAC after the CPU is started-up.

Preferably, the DMAC and/or the start-up logic circuit is/are provided on the same chip as the CPU.

In a second aspect, the invention provides a method of starting-up a processor system, the method comprising the steps of:

receiving a start-up signal by a start-up logic circuit;

enabling a DMAC by the start-up logic circuit;

accessing a non-volatile storage device by the DMAC to load at least initial operating code for the CPU into a volatile RAM;

enabling the CPU to access the volatile RAM, thereby initiating start-up of the CPU.

Brief Description Of The Drawings

Various embodiments of the invention will now be more fully described, by way of example, with reference to the drawings, of which:

FIG. 1 shows a schematic block diagram of a first prior art processor system

FIG. 2 shows a schematic block diagram of a second prior art processor system

FIG. 3 shows a schematic block diagram of a processor system according to an embodiment of the invention.

Detailed Description

Thus, as shown in FIG. 1 and as described briefly above, in a first prior art system a CPU 3 is coupled to receive start-up signals, indicated by arrow 2, from a Power-On Circuit 1. The CPU 3 is also coupled to receive signals, via line 4, from an EPROM 5, or any other NVM, and signals, via line 13, from a RAM 8 and to send signals, via line 9, to control a DMAC 6. The DMAC 6 sends signals via line 14 to control a Non Volatile Storage Device (NVSD) 7 to output information to the RAM 8, from where it can be read by the CPU 3. When power is first applied, the Power-On Circuit 1 sends a start-up signal via line 2 to an input of the CPU 3. The CPU 3 responds to this by fetching code from a pre-defined "address" that is mapped to a location in EPROM 5. The pre-defined address that is accessed by the CPU 3 following the startup signal from the Power-On Circuit 1 is determined by the design of the CPU 3.

The power-on sequence continues with the CPU 3 executing software held in the EPROM 5. A portion of this code causes the CPU 3 to program the DMAC 6 to transfer operating software from the NVSD 7 to the RAM 8.

This involves the CPU 3 writing into registers in the DMAC 6 various parameters that describe the NVSD 7, the destination address, and the mode of operation. When the operating software held in NVSD 7 is available and access to RAM 8 is not blocked, DMAC 6 starts the transfer.

It will be appreciated that some NVSDs (such as discs) may need to be programmed before the operating software can be transferred. The software can either be transferred directly from the NVSD 7 to the RAM 8 via line 10, or first transferred from the NVSD 7 to the DMAC 6 via line 11 and then from the DMAC 6 to the RAM 8 via line 12. Following the transfer of operating software, the processor CPU 3 can excute code held in the RAM 8, thus allowing the system to function. Of course, although the CPU 3, EPROM 5, RAM 8 and DMAC 6 are shown as separate blocks, any combination of these may be integrated for manufacture. As mentioned above, one disadvantage of this system is that the EPROM 5 is expensive and is usually of smaller data width than the CPU 3, which consequently requires longer time to access all the operating code than if the EPROM 5 had the same data width as the CPU 3.

In the second prior art system described briefly above and shown in FIG. 2 with the same elements as described with reference to FIG. 1 having the same reference numerals, after power has been applied, the Power-On Circuit 1 provides a start-up signal 2 to an input of a second CPU 16. At this time, immediately after power has been applied, the second CPU 16 sends a signal on line 20 to a Reset Control Circuit 15, which provides a signal 17 to keep the reset input of CPU 3 asserted. This prevents the CPU 3 from running until initial operating code has been placed in the volatile RAM 8.

The power-on sequence continues with the second CPU 16 receiving, via line 18, software held in the EPROM 5. A portion of this code causes the second CPU 16 to program, via line 19, the DMAC 6 to control, via line 14, the NVSD 7 to transfer operating software from the NVSD 7 to RAM 8. As in the first prior art system described above, this involves second CPU 16 writing into the DMAC 6 parameters that describe the NVSD 7, the destination address, and the mode of operation. The destination address corresponds to the location in the RAM 8 from which CPU 3 starts fetching the software via line 13 after it's reset input has been negated. As before, the software can either be transfered directly from NVSD 7 to the RAM 8, via line 10, or transferred from the NVSD 7 to the DMAC 6 via line 11 and then from the DMAC 6 to the RAM 8 via line 12. Following the transfer of

operating software, the second CPU 16 signals, via line 20, the Reset Control Circuit 14 which responds by negating, via line 17, the reset input of CPU 3 allowing it to run. Now CPU 3 is able to receive, via line 13, and execute the code held in the RAM 3 thus allowing the system to function.

As mentioned above, the disadvantage of this system is that it requires a second CPU, which is clearly expensive, and does not solve the problem of how that second CPU is first started up.

FIG. 3 shows an embodiment of a system according to the present invention in which identical elements to those of the prior art systems described above with reference to FIGS. 1 and 2 have the same reference numerals. In this embodiment, the system includes a Central Processing Unit (CPU) 3 that executes instructions held in memory, a volatile Random Access Memory (RAM) 8 that loses it's contents on power down, a Direct Memory Access Controller (DMAC) 25 that performs a data transfer automatically on system power-up, a Non-Volatile Storage Device (NVSD) 7 that holds operating software for the system (and can supply data for the power-on DMAC-based transfer without the intervention of CPU 3) and a Power-On Circuit 1 that controls the reset inputs of CPU 3 and DMAC 25.

After power has been applied, Power-On Circuit 1 sends a start-up signal on line 2 to keep the reset input of CPU 3 asserted. This prevents CPU 3 from running until initial operating code has been placed in the volatile RAM 8. After power has been applied, Power-On Circuit 1 also provides a signal on line 21 to negate a reset input of the DMAC 25. This causes three Power-on values POV1, POV2 and POV3 to be loaded from Non-Volatile Memory Addresses 28, 31 and 34, respectively, in the DMAC 25 via respective connections 27, 30 and 33 into three operational registers 26, 29 and 32, respectively. These operational registers 26, 29 and 32 control the execution of a DMA transfer. The Power-on values POV1, POV2 and POV3 could alternatively be hard-wired in the DMAC, for example in registers.

The Power-on values POV1, POV2 and POV3 specify the parameters for the first DMA transfer that occurs, without CPU intervention, on power up. Operational register 26 controls the destination address for the transfer and value POV1 specifies the destination address for the power-up transfer, which corresponds to the location in the RAM 8 that CPU 3 attempts to fetch code from after it's reset input is negated. Operational register 29 describes the source device for a given transfer and value POV2 provides a description of the NVSD 7 for the power-up transfer.

Operational register 32 controls the remaining aspects of the transfer and includes fields that specify the transfer quantity and the transfer start command. Value POV3 specifies a transfer quantity that will allow the system to boot and it sets the field corresponding to the start command. Once registers 26, 29 and 32 are loaded with the respective values POV1, POV2 and POV3, the DMAC 25 starts the transfer. The software can either be transferred directly from NVSD 7 to the RAM 8 via line 10, or firstly transferred from NVSD 7 to DMAC 25 via line 11 and then from DMAC 25 to RAM 8 via line 12, as described above with reference to FIGS. 1 and 2.

Following the transfer of operating software, DMAC 25 sends a signal via line 22 to the Power-On Circuit 1 which responds by negating the reset input of CPU 3 via line 2. Following the negation of the reset input, the CPU 3 fetches code via line 23 from a predefined address in RAM 8 that maps to the destination location of the DMAC's power-on transfer. Following this inital transfer, DMAC 25 is programmed, via line 24, by CPU 3 to perform additional transfers, if required.

Thus, it is clear that in this system, there is no requirement for a separate, and expensive, EPROM, as in the first prior art system described above, nor for a second CPU, as in the second prior art system described above. Instead, the DMAC is hard wired to provide sufficient commands for it to be able to control the NSVD to transfer sufficient operating software to the RAM to enable the CPU to start operating.

It will be appreciated that although the system components are shown as separate blocks, any combination of these may be integrated for manufacture. Furthermore, some NVSDs (typically discs) require to be programmed before they can support a data transfer. This can be achieved by using default power-on values without requiring the intervention of CPU 3.

It will be appreciated that although only one particular embodiment of the invention has been described in detail, various modifications and improvements can be made by a person skilled in the art without departing from the scope of the present invention. For example, other embodiments of the invention may allow the CPU 3 to access and execute the uninitialised contents of the RAM 8. Once code has been transferred from the NVSD 7 an event can be used to re-direct the CPU 3 to the transfer destination. Furthermore, although this embodiment shows a simple DMAC with a single set of registers, other embodiments may be more

complex and support additional features. It will also be appreciated that although the reset inputs of the CPU 3 and the DMAC 25 are used to control the power-up sequence, other signals (such as bus arbitration signals) may be used.

Claims

1. A processor system comprising a Central Processing Unit (CPU), a Direct Memory Access Controller (DMAC) coupled to the CPU, a non-volatile storage device for storing at least initial operating code, volatile Random Access Memory (RAM) coupled to the CPU, and a start-up logic circuit coupled to the DMAC , the start-up logic circuit enabling the DMAC when a start-up signal is received, the DMAC being configured to access the non-volatile data storage device to load at least the initial operating code for the CPU into the volatile RAM and, when at least part of the initial operating code is loaded in the volatile RAM, to enable the CPU to start to access at least part of the initial operating code from the volatile RAM, thereby initiating start-up of the CPU.

2. A processor system according to claim 1, wherein the start-up logic circuit is coupled to the CPU to restrain the CPU from accessing the volatile RAM when a start-up signal is received until the DMAC enables the CPU.

3. A processor system according to either claim 1 or claim 2, wherein the DMAC enables the CPU directly.

4. A processor system according to either claim 1 or claim 2, wherein the start-up logic circuit enables the CPU when it receives a signal from the DMAC.

5. A processor system according to any preceding claim, wherein the non-volatile storage device is a Non-Volatile Memory (NVM).

6. A processor system according to any one of claims 1 to 4, wherein the non-volatile storage device is chosen from a hard file, such as a disc, CD,or optical CD, or other storage medium, such as magnetic tape.

7. A processor system according to any preceding claim, wherein the DMAC accesses the non-volatile storage device by a direct link.

8. A processor system according to any preceding claim, wherein the DMAC accesses the non-volatile storage device by indirect data transfer, such as wireless transfer, e.g. by radio, infra-red, or other means.

9. A processor system according to any preceding claim, wherein the non-volatile storage device has a different data width to that of the CPU and the DMAC reformats the initial operating code into the same data width as the CPU before it is loaded into the volatile RAM.

10. A processor system according to any preceding claim, wherein the non-volatile storage device stores complete operating software for the CPU, as well as the initial operating code.

11. A processor system according to any one of claims 1 to 9, wherein the operating software is stored on other non-volatile storage devices, which are accessed by the DMAC after the CPU is started-up.

12. A processor system according to any preceding claim, wherein the DMAC is provided on the same chip as the CPU.

13. A processor system according to any preceding claim, wherein the start-up logic circuit is provided on the same chip as the CPU.

14. A method of starting-up a processor system, the method comprising the steps of:

receiving a start-up signal by a start-up logic circuit;

enabling a DMAC by the start-up logic circuit;

accessing a non-volatile storage device by the DMAC to load at least initial operating code for the CPU into a volatile RAM;

enabling the CPU to access the volatile RAM, thereby initiating start-up of the CPU.

15. A method of starting-up a processor system according to claim 14, further comprising the step of:

restraining the CPU from accessing the volatile RAM when a start-up signal is received until the DMAC enables the CPU.

16. A method of starting-up a processor system according to either claim 14 or claim 15, further comprising the step of:

reformatting the initial operating code into the same data width as the CPU before it is loaded into the volatile RAM when the non-volatile storage device has a different data width to that of the CPU and the DMAC.

17. A processor system substantially as hereinbefore described with reference to FIG. 3 of the drawings.

18. A method of starting-up a processor system substantially as hereinbefore described with reference to FIG. 3 of the drawings.

The Patent Office

10

Application No: GB 9516082.6
Claims searched: 1-18

Examiner: B.G.Western
Date of search: 19 October 1995

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.N): G4A AFL

Int Cl (Ed.6): G06F 9/24 9/445

Other: On-line : WPI, INSPEC, COMPUTER

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
X	GB-2184577-A MITEL N.b pages 2-4	1-3,6,7, 10,13-15
A	EP-0100140-A2 DATA GENERAL See whole document	-

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.